**EXPERIMENT 1:**

**To implement any IR modelling technique**

Zoya Momin

Department of Computer Engineering

M.H Saboo Siddik College of Engineering

Mumbai, India

zoya.221257.co@mhssce.ac.in

**I.**  **INTRODUCTION**

Information Retrieval (IR) modeling techniques are methods used to rank documents based on their relevance to a user's query. These techniques involve representing documents and queries in a structured way and then using a ranking function to determine the order in which results are presented to the user.  The system assists users in finding the information they require but it does not explicitly return the answers of the questions. It informs the existence and location of documents that might consist of the required information. The documents that satisfy users requirement are called relevant documents. A perfect IR system will retrieve only relevant documents.

The **Vector Space Model (VSM)** is a widely used IR model where both documents and queries are represented as vectors in a high-dimensional space. Each dimension corresponds to a unique term in the corpus, and the weight of each term can be computed using different schemes like **Term Frequency (TF)**, **Inverse Document Frequency (IDF)**, or **TF-IDF**.

* **Term Frequency (TF):** Measures how frequently a term appears in a document.
* **Inverse Document Frequency (IDF):** Measures how important a term is, reducing the weight of common terms.
* **TF-IDF:** Combines TF and IDF to give higher scores to important but less frequent words.
* **Cosine Similarity:** Measures the cosine of the angle between two vectors to determine document-query similarity.

This model is highly effective for text retrieval as it allows ranking documents by relevance instead of simple keyword matching.

II. STEPS

**Step 1: Text Corpus Preparation & Pre-processing**

1. Collect a small text corpus (10–20 .txt files) on a chosen theme (e.g., technology, health, education).
2. Convert all text to lowercase.
3. Remove punctuation, numbers, and special characters.
4. Tokenize the text into words.
5. Remove stop words (common words like “is”, “the”, “and”).
6. Perform stemming or lemmatization to reduce words to their root form.
7. Store the cleaned tokens for each document in a structured format (dictionary or DataFrame).

Step 2: Term Weight Calculation (TF-IDF)

1. **TF Calculation:** Count term frequency for each word in each document.
2. **IDF Calculation:** Use the formula: IDF(t)=log(N/dft)  
   where *N* = total number of documents, *df\_t* = number of documents containing term *t*.
3. Multiply TF and IDF values to get TF-IDF weights.
4. Implement using TfidfVectorizer from sklearn.

**Step 3: Query Input and Vectorization**

1. Create a simple interface for user query input (command-line or GUI).
2. Preprocess the query using the same steps as the documents.
3. Vectorize the query using the fitted TF-IDF model.

**Step 4: Similarity Calculation**

1. Represent all documents and the query in the same TF-IDF vector space.
2. Use cosine similarity to calculate similarity scores between the query and each document.
3. Store similarity scores in a list or DataFrame.

**Step 5: Ranking & Retrieval**

1. Sort documents in descending order of similarity score.
2. Display the top-*k* most relevant documents.
3. Show title/snippet and similarity score for each result.

**Step 6: Evaluation**

1. Prepare a set of queries with known relevant documents (ground truth).
2. Compute **Precision**, **Recall**, and **F1-score** for each query.
3. Calculate Mean Average Precision (MAP) across multiple queries.

**Step 7: Experimentation & Observation**

1. Try different term weighting schemes (binary, TF, TF-IDF).
2. Apply document length normalization.
3. Test query expansion with synonyms.
4. Record how retrieval performance changes for each variation.

